COMP3411/9814 Artificial Intelligence
Term 1, 2024

Assignment 2 – Heuristics and Search

Due: Tuesday 2nd April, 2pm
Marks: 12% of final assessment

In this assignment you will be examining search strategies for the 15-puzzle, admissible heuristics for a trajectory planning problem, and pruning in alpha-beta search trees. You should submit a report in .pdf format with answers to the questions below.

Question 1: Search Strategies for the 15-Puzzle (2 marks)

For this question you will construct a table showing the number of states expanded when the 15-puzzle is solved, from various start positions, using four different search strategies:

(i) Breadth First Search
(ii) Iterative Deepening Search
(iii) Greedy Search (using the Manhattan Distance heuristic)
(iv) A* Search (using the Manhattan Distance heuristic)

Download the file `path_search.zip` from this directory:
`https://www.cse.unsw.edu.au/~cs3411/24T1/code/`
Unzip the file and change directory to `path_search`

```
unzip path_search.zip
cd path_search
```

Run the code by typing:

```
python3 search.py --start 2634-5178-AB0C-9DEF --s bfs
```

The `--start` argument specifies the start position, which in this case is:

2	6	3	4
5	1	7	8
A	B		C
9	D	E	F

Start State

1	2	3	4
5	6	7	8
9	A	B	C
D	E	F	

Goal State

The Goal State is shown on the right. The `--s` argument specifies the search strategy (`bfs` for Breadth First Search).

The code should print out the number of expanded nodes (by thousands) as it searches. It should then print a path from the Start State to the Goal State, followed by the number of nodes Generated and Expanded, and the Length and Cost of the path (which are both equal to 12 in this case).

(a) Draw up a table in this format:

Start State	BFS		IDS		Greedy		A*	
start1 start2 start3								

Run each of the four search strategies from three specified start positions, using the following combinations of command-line arguments:

Start Positions:

start1: `--start 2634-5178-AB0C-9DEF`
start2: `--start 1034-728B-5D6A-E9FC`
start3: `--start 5247-61C0-9A83-DEBF`

Search Strategies:

BFS: `--s bfs`
IDS: `--s dfs --id`
Greedy: `--s greedy`
A*Search: `--s astar`

In each case, record in your table the length of the path, and the number of nodes Expanded during the search. Include the completed table in your report.

(b) Briefly discuss the efficiency of these four search strategies, with regard to the number of nodes Expanded, and the length of the resulting path.

Question 2: Heuristic Path Search for 15-Puzzle (2 marks)

In this question you will be exploring a search strategy known as **Heuristic Path Search**, which is a best-first search using the objective function:

$$f_w(n) = (2 - w)g(n) + wh(n),$$

where w is a number between 0 and 2. Heuristic Path Search is equivalent to Uniform Cost Search when $w = 0$, to A* Search when $w = 1$, and Greedy Search when $w = 2$. It is Complete for all w between 0 and 2.

(a) Prove that Heuristic Path Search is **optimal** when $0 \leq w \leq 1$, assuming $h()$ is admissible. (You may also assume that A* Search is optimal.)

Hint: show that minimizing $f_w(n) = (2 - w)g(n) + wh(n)$ is the same as minimizing $f'_w(n) = g(n) + h'(n)$ for some function $h'(n)$ with the property that $h'(n) \leq h(n)$ for all n.

(b) Draw up a table in this format (the top row has been filled in for you):

	start4		start5		start6	
IDA* Search	45	545120	50	4178819	56	169367641
HPS, $w = 1.1$					58	13770561
HPS, $w = 1.2$						
HPS, $w = 1.3$						
HPS, $w = 1.4$						

Run `search.py` on each of the three start states shown below, using the Iterative Deepening version of Heuristic Path Search, with $w = 1.1,\ 1.2,\ 1.3$ and 1.4 .

Start Positions:

start4: `--start A974-3256-FD8B-EC01`
start5: `--start 153E-A02C-9FBD-8476`
start6: `--start 418E-7AD0-9C52-3FB6`

Search Strategies:

HPS, $w = 1.1$: `--s heuristic --w 1.1 --id`
HPS, $w = 1.2$: `--s heuristic --w 1.2 --id`
HPS, $w = 1.3$: `--s heuristic --w 1.3 --id`
HPS, $w = 1.4$: `--s heuristic --w 1.4 --id`

In each case, record in your table the length of the path, and the number of nodes Expanded during the search. Include the completed table in your report.

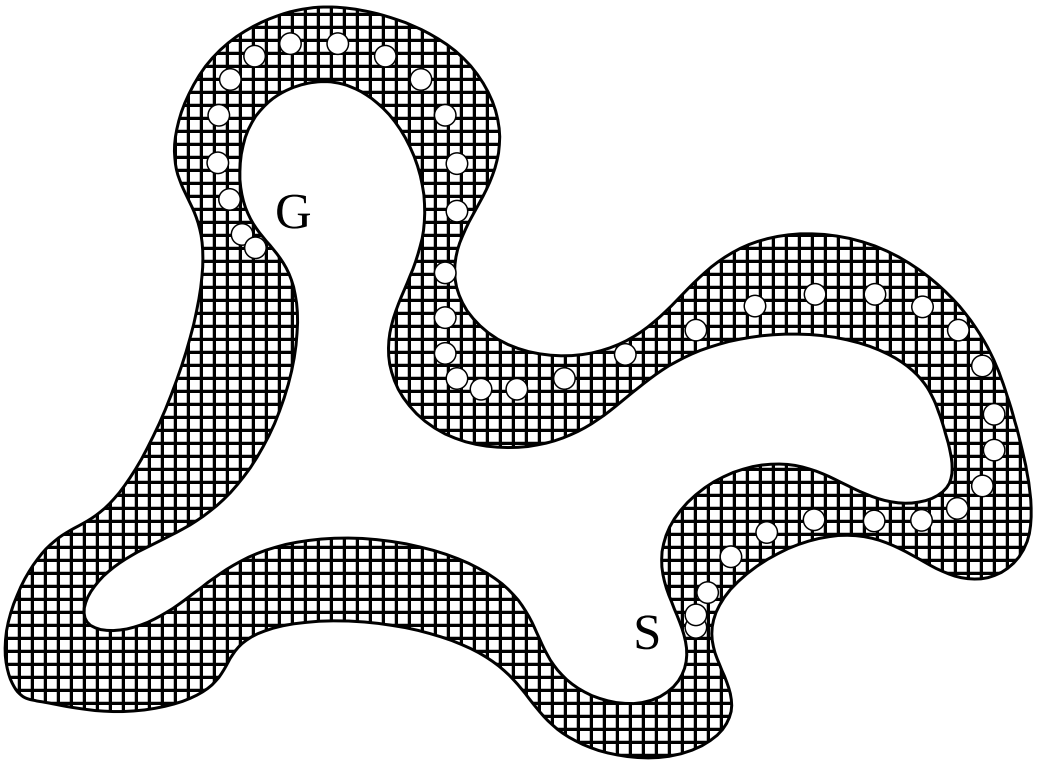
(c) Briefly discuss how the path Length and the number of Expanded nodes change as the value of w varies between 1.0 (equivalent to IDA*) and 1.4.

Question 3: Graph Paper Grand Prix (5 marks)

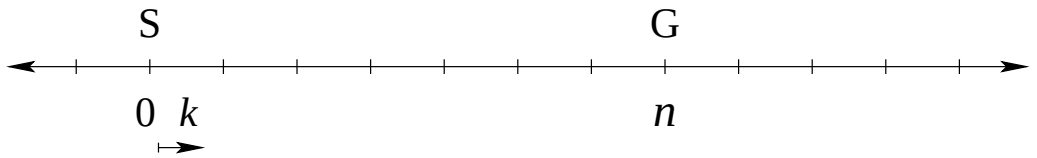
We saw in lectures how the Straight-Line-Distance heuristic can be used to find the shortest-distance path between two points. However, in many robotic applications we wish to minimize not the distance but rather the **time** taken to traverse the path, by speeding up on the straight bits and avoiding sharp corners.

In this question you will be exploring a simplified version of this problem in the form of a game known as Graph Paper Grand Prix (GPGP).

To play GPGP, you first need to draw the outline of a racing track on a sheet of graph paper, and choose a start location S = (r_S, c_S) as well as a Goal location G = (r_G, c_G) where r and c are the row and column. The agent begins at location S, with velocity (0,0). A "state" for the agent consists of a position (r, c) and a velocity (u, v), where r, c, u, v are (positive or negative) integers.



At each time step, the agent has the opportunity to increase or decrease each component of its velocity by one unit, or to keep it the same. In other words, the agent must choose an acceleration vector (a, b) with $a, b \in \{-1, 0, +1\}$. It then updates its velocity from (u, v) to $(u', v') = (u + a, v + b)$, and updates its position – using the *new* velocity – from (r, c) to $(r + u', c + v')$. The aim of the game is to travel as fast as possible, but without crashing into any obstacles or running off the edge of the track, and eventually stop at the Goal with velocity $(0, 0)$.



We first consider a 1-dimensional version of GPGP where the vehicle moves through integer locations on a number line, with no obstacles. Assume the Goal is at location n, and that the agent starts at location 0, with velocity k. We will use $M(n,k)$ to denote the minimum number of time steps required to arrive and stop at the Goal. Clearly $M(-n,-k) = M(n,k)$ so we only need to compute $M(n,k)$ for $k \geq 0$.

(a) Starting with the special case $k = 0$, compute $M(n,0)$ for $1 \leq n \leq 21$ by writing down the optimal sequence of actions for all n between 1 and 21. For example, if $n = 7$ then the optimal sequence is $[+ + \circ - \circ -]$ so $M(7,0) = 6$. (When multiple solutions exist, you should pick the one which goes "fast early" i.e. with all the +'s at the beginning.)

(b) Assume $n \geq 0$. By extrapolating patterns in the sequences from part (a), explain why the general formula for $M(n,0)$ is

$$M(n,0) = \left\lceil 2\sqrt{n} \right\rceil,$$

where $\lceil z \rceil$ denotes z rounded up to the nearest integer.
Hint: Do not try to use recurrence relations. You should instead use this identity:

$$\left\lceil 2\sqrt{n} \right\rceil = \begin{cases} 2s+1, & \text{if} \quad n = s^2 + j, & 1 \leq j \leq s \\ 2s+2, & \text{if} \quad n = s^2 + s + j\,, & 1 \leq j \leq s \\ 2s+2, & \text{if} \quad n = (s+1)^2 & \end{cases}$$

(c) Assuming the result from part (b), show that if $k \geq 0$ and $n \geq \frac{1}{2}k(k-1)$ then

$$M(n,k) = \left\lceil 2\sqrt{n + \tfrac{1}{2}k(k+1)} \right\rceil - k$$

Hint: Consider the path of the agent as part of a larger path.

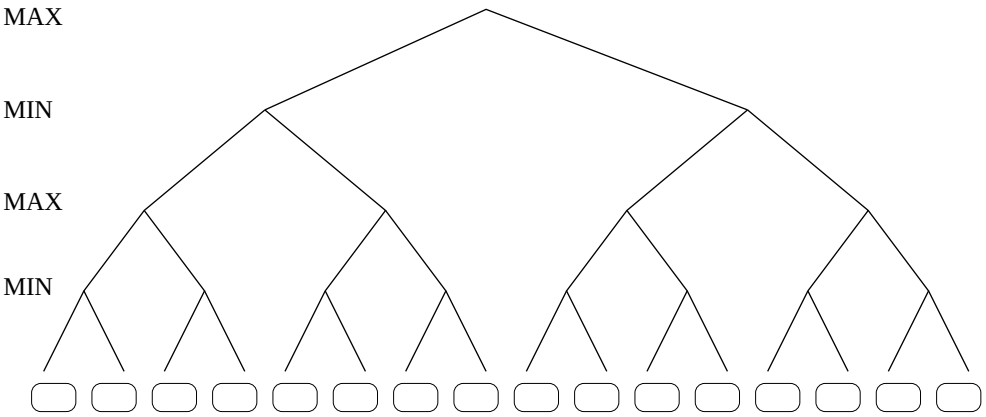
(d) Derive a formula for $M(n,k)$ in the case where $k \geq 0$ and $n < \frac{1}{2}k(k-1)$.

(e) Write down an admissible heuristic (that approximates the true number of moves as closely as possible) for the original 2-dimensional GPGP game in terms of the function $M()$ derived above.
Hint: Consider the horizontal and vertical motion separately, keeping in mind that, unlike the 15-puzzle, the agent can move in the horizontal and vertical direction simultaneously. Your heuristic should be of this form:

$$h\,(r, c, u, v, r_G, c_G) = \max(M(..,..), M(..,..))$$

Question 4: Game Trees and Pruning (3 marks)

(a) Consider a game tree of depth 4, where each internal node has exactly **two** children (shown below). Fill in the leaves of this game tree with all of the values from 0 to 15, in such a way that the alpha-beta algorithm prunes as many nodes as possible. Hint: make sure that, at each branch of the tree, all the leaves in the left subtree are preferable to all the leaves in the right subtree (for the player whose turn it is to move).



(b) Trace through the alpha-beta search algorithm on your tree, clearly showing which of the original 16 leaves are evaluated.

(c) Now consider another game tree of depth 4, but where each internal node has exactly **three** children. Assume that the leaves have been assigned in such a way that the alpha-beta algorithm prunes as many nodes as possible. Draw the shape of the pruned tree. How many of the original 81 leaves will be evaluated?

Hint: If you look closely at the pruned tree from part (b) you will see a pattern. Some nodes explore all of their children; other nodes explore only their leftmost child and prune the other children. The path down the extreme left side of the tree is called the line of best play or Principal Variation (PV). Nodes along this path are called PV-nodes. PV-nodes explore all of their children. If we follow a path starting from a PV-node but proceeding through non-PV nodes, we see an alternation between nodes which explore all of their children, and those which explore only one child. By reproducing this pattern for the tree in part (c), you should be able to draw the shape of the pruned tree (without actually assigning values to the leaves or tracing through the alpha-beta algorithm).

(d) What is the time complexity of alpha-beta search, if the best move is always examined first (at every branch of the tree)? Explain why.

Submission

This assignment must be submitted electronically – either through WebCMS, or from the command line, using:

```
give cs3411 hw2 hw2.pdf
```

The submission deadline is Tuesday 2nd April, 2pm. Late submissions will incur a penalty of 5% per day, up to a maximum of 5 days.

Group submissions will not be allowed. By all means, discuss the assignment with your fellow students. But you must write (or type) your answers individually. Do NOT copy anyone else's assignment, or send your assignment to any other student.

Good luck!